Design:

- FlickrRequestable:
    - This is a Service protocol. Any class that conforms to this protocol can define custom parameters around populating a request (i.e. request headers, url parameters, etc).

- FlickrSearchResult:
    - This is the response object (from the provided API).
    - One enhancement to this Model that I would suggest is to have this object conform to the Decodable protocol. If you were to peek in the FlickrSearchFactory class, I manually parse the response and map it to FlickrSearchResult (this can be done much easily with the use of the Decodable protocol).

- FlickrSearchFactory:
    - This is the model object that is responsible for making API calls, pagination, and performing the search queries. I have also implemented an extremely light weight cache (String:Data dictionary).
    - If you were to take a look at the successCompletionBlock (part of conforming to the FlickrRequestable protocol), you will notice that I am filtering out non “jpg” and “png” types.

- FlickrCollectionViewCell:
    - CollectionViewCell that has an ImageView and a Title

- FlickrSearchView:
    - Search View that contains a Search Bar and a CollectionView

- FlickrSearchViewController:
    - The biggest challenge here was to ensure that if we are in the process of fetching Image data on a background thread, but the cell is no longer on the screen and possible reused, you do not need to update the cell with the Image (just cache it instead).
    - 250 ms debounce -> The timer object helps us with the debounce.

- Other Comments / Logic behind decisions:
    - I decided to use a non-storyboard approach because I wanted to show my understanding of constraints and how they work.